Isolated and Exhausted: Attacking Operating Systems via Site Isolation in the Browser

Matthias Gierlings, Marcus Brinkmann, and Jörg Schwenk, *Ruhr University Bochum*

https://www.usenix.org/conference/usenixsecurity23/presentation/gierlings

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

USENIX'23 Artifact Appendix: **Isolated and Exhausted: Attacking Operating Systems via Site Isolation in the Browser**

Matthias Gierlings, Marcus Brinkmann, Jörg Schwenk
Ruhr University Bochum

A Artifact Appendix

A.1 Abstract

Site Isolation, a security feature recently introduced to major browsers enables attacks on modern operating systems. To demonstrate the impact of Site Isolation attacks on web users we implemented a Site Isolation fork-bomb and a DNS Cache Poisoning Attack: *DNS Poisoning by Exhaustive Misappropriation of Network Sockets (DEMONS)*. Setup instructions, configurations, and the implementation of both attacks are part of our publicly available research artifacts. While DEMONS was assigned CVE-2020-6557 and patched by the Chromium Team,[1] the fork-bomb is still a threat to current browsers. We describe a way to mitigate the Site Isolation fork-bomb in Chromium-based browsers without measurable performance penalty and include both the patch and our performance measurement results in our artifacts.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

All artifacts provided should be evaluated in a strictly isolated, physical or virtual lab environment. In case reviewers decide to evaluate attacks using public internet infrastructure they must employ measures to prevent *any* traffic from flowing between systems that are part of their evaluation setup and other systems.

A.2.2 How to access

The artifact repository is available at `https://git.noc.rub.de/gierlmds/isolated-and-exhausted/-/tree/usenix23_ae_summer`. Additionally we provide a mirror-copy[2] via Zenodo.

A.2.3 Hardware dependencies

Our experiments can be conducted either in a VM-based lab environment on a single x86 machine (VM setup) or alternatively on a set of physical hosts (hardware setup). In both cases, the setup consists of four distinct hosts (physical or virtual). For the VM setup an x86 desktop system with at least 4 cores/8 threads, virtualization support[3], at least 20 GiB RAM and more than 400 GiB of free disk space is required.

DEMONS VM Setup We recommend using Virtual Box 6[4] for virtualization and Kubuntu Linux 22.04 LTS as host OS.

DEMONS Hardware Setup For our hardware setup we used three desktop computers[5] to run the victim, the benign DNS server and the router. A ThinkPad T480s[6] was used to run the attacker infrastructure (webserver and spoofer).

Fork-Bomb Hardware Setup The fork bomb was evaluated on a Dell Latitude 5280, Intel Core i5 7200U, 8 GiB RAM, 240 GiB M.2 SATA SSD running Kubuntu Linux 18.04.5 LTS (Kernel 5.4.0-62).

A.2.4 Software dependencies

In addition to our artifacts additional software (binaries/installers) are required. This section lists the exact software versions used in testing our artifacts, their use is strongly recommended. While we did not verify that other versions than those listed below produce the same results, we believe that our artifacts will work with any version of Windows 10 and any release of Chrome with Site Isolation Support prior to version 86.0.4240.75, which introduces a fix for CVE-2020-6557. Using newer versions of Python 3 and the Python Websocket Client are unlikely to impact the experiments. Using different Virtual Box versions or other Hypervisors should also be possible but potentially affects performance and may require manual timing adjustments in the code.

- Windows 10 (1909 Build 18363.815)
- Ubuntu Linux 20.04.5 LTS Server
- Kubuntu Linux 18.04.5 LTS (Kernel 5.4.0-62)
- Kubuntu Linux 20.04.5 LTS
- Kubuntu Linux 22.04.1 LTS
- Chrome 83.0.4103.106 (Windows)

[1] https://chromereleases.googleblog.com/2020/10/stable-channel-update-for-desktop.html
[2] DOI: 10.5281/zenodo.7356538

[3] Intel VT-x or AMD-V
[4] 6.1.38-dfsg-3 ubuntu1.22.04.1 amd64
[5] Intel Core2Quad Q9400, 4 GiB RAM, Intel 82567LM-3 Gigabit NIC
[6] Intel i7-8550U, 40 GiB RAM, Intel Ethernet I219-V

- Firefox Nightly 86.01a (Windows, Linux)
- Chromium 83.0.4103.0 (Linux)
- Caddyserver 1.0.4[7] (Linux)
- Python 3.8.2 (Windows)
- Python Websocket Client for Python 3.8.2 (Windows)
- Virtual Box 6[4] (Linux)

It is recommended to download all binary dependencies except Caddyserver[7] before starting the setup procedure and transfer them via shared folders in case of a VM setup or via USB stick in case of a setup with physical hosts. This way the required software can be installed without internet connection. This also prevents unsolicited automatic browser/OS updates.

In addition to our artifacts the following software source code is required:

- Chromium 101.0.4951.64[7] (Linux)

A.2.5 Benchmarks

To benchmark the performance impact of our Site Isolation mitigation we used the performance profiler[8] integrated into the Chrome developer tools. The profiler can be accessed via the *Performance*-tab.

A.3 Set-up

A.3.1 Installation

The lab environment for the Site Isolation fork-bomb and DEMONS consists of four hosts, the victim, a router, a benign DNS server, and the attacker's server. This section contains detailed instructions on how to set up those hosts as virtual machines using Virtual Box. The instructions are also suitable for installation on native hardware. In this case, Virtual Box-specific steps should be omitted.

Base System The base image serves as a common base installation for the router, the benign DNS and the attacker server. To prepare the base image, perform the following steps:

1. Install Oracle VirtualBox.[9]
2. Create a new 64-bit Ubuntu Linux VM with 2 cores and 2 GiB RAM and 15 GiB hard disk.
3. Enable the following acceleration settings: `VT-x/AMD-V`, `Nested Paging`, `PAE/NX`, `KVM Paravirtualization`
4. Install the Ubuntu 20.04 LTS server base image. This guide and the derivative guides assume that during the installation process `user` was chosen as username and `si-base` was chosen as the hostname.
5. Update the OS and install additional packages:

```
sudo apt update && sudo apt dist-upgrade && sudo
    ↪ apt autoremove --purge
sudo apt install build-essential curl git iptables
    ↪ -persistent
```

6. Insert the VirtualBox guest additions ".iso" into the virtual CD-ROM drive of your machine.
7. Install the VirtualBox guest additions.

```
sudo mount /dev/sr0 /media
cd /media
sudo ./VBoxLinuxAdditions.run
```

8. Add `user` to the `vboxsf` group

```
sudo usermod -a -G vboxsf user
```

9. Set an environment variable referring to the artifact repository base folder. It should be located at: `/home/user/isolated-and-exhausted`

```
export ARTIFACTS_REPO="/home/user/isolated-and-
    ↪ exhausted/"
sudo bash -c "echo ARTIFACTS_REPO=${ARTIFACTS_REPO
    ↪ } >> /etc/environment"
```

10. Clone the Isolated and Exhausted artifacts repository:

```
git clone https://git.noc.ruhr-uni-bochum.de/
    ↪ gierlmds/isolated-and-exhausted
    ↪ $ARTIFACTS_REPO
```

11. Shut down the system.

```
sudo systemctl poweroff
```

Attacker Server The attacker server runs the attacker's web server and the spoofer. Both the web server and the spoofer are containerized using Docker. To set up the attacker server perform the following steps on top of a *Base System* (cf. section A.3.1 Base System):

1. Clone the base system to a new virtual or physical host.
2. Name the cloned VM `Isolated and Exhausted Attacker`.
3. Boot the system and change the hostname to `si-attacker`:

```
sudo bash -c "echo si-attacker > /etc/hostname"
```

4. Install docker.

```
curl -fsSL https://download.docker.com/linux/
    ↪ ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://
    ↪ download.docker.com/linux/ubuntu $(
    ↪ lsb_release -cs) stable"
sudo apt update
sudo apt install docker-ce docker-ce-cli
    ↪ containerd.io docker-compose
```

[7] Will be obtained during setup (cf. subsection A.3)
[8] https://developer.chrome.com/docs/devtools/evaluate-performance/
[9] https://www.virtualbox.org/

5. Add `user` to the docker group.

```
sudo usermod -a -G docker $USER
```

6. Get the ubuntu base image from Docker Hub.

```
sudo docker pull ubuntu:bionic-20200311
```

7. Copy the attacker host configuration[10]

```
sudo rm -f /etc/netplan/*
sudo cp -R $ARTIFACTS_REPO/hosts/attacker/rootfs/*
    ↪ /
```

8. Download a copy of the caddy web server into the repository. For our evaluation caddy version 1.0.4 was used. Newer caddy versions should work but may require modifications to Caddyfiles.

```
cd $ARTIFACTS_REPO/hosts/attacker/webserver/
curl -JLO 'https://github.com/caddyserver/caddy/
    ↪ releases/download/v1.0.4/caddy_v1.0.4
    ↪ _linux_amd64.tar.gz'
```

9. Replace `/etc/resolve.conf` with a sym-link:

```
sudo ln -sf /var/run/systemd/resolve/resolv.conf /
    ↪ etc/resolv.conf
```

10. Add the base folder for docker volumes.

```
sudo mkdir -p /srv/docker/data/demons/attacker/
    ↪ spoofer/logs /srv/docker/data/demons/
    ↪ attacker/spoofer/results
sudo chown -R root:docker /srv/docker
sudo chmod -R 775 /srv/docker
```

11. Enable IPv6 forwarding and redirect all traffic on the attacker's subnet `2001:db8::/113` to the web server running on `2001:db8::1`.

```
sudo sysctl -w net.ipv6.conf.all.forwarding=1
sudo ip6tables -t nat -A PREROUTING -d 2001:db8
    ↪ ::/113 -j DNAT --to-destination 2001:db8
    ↪ ::1
```

12. Spoof the spoofers' source IP address.

```
sudo ip6tables -t nat -A POSTROUTING -s 2001:db8
    ↪ ::8001 -p udp -j SNAT --to-source 2001:db8
    ↪ ::8000:1
```

13. Make iptables rules persistent after reboot.

```
sudo netfilter-persistent save
```

14. Shut down the system.

```
sudo poweroff
```

15. Change the VM network settings
    - Right-click the Attacker VM in the VirtualBox Manager
    - Select the option `Settings` from the drop-down menu.
    - Make sure that `Adapter 1` and `Adapter 2` are enabled and that all other adapters are disabled
    - Attach `Adapter 1` to an `Internal Network` named `attacker_provider`. Set the adapter type to `Paravirtualized Network (virtio-net)`
    - For `Adapter 2` set the network type to `NAT`
16. Boot the system.
17. Create docker containers containing the attacker web server and spoofer.

```
cd $ARTIFACTS_REPO/hosts/attacker/
docker-compose up -d --build
```

18. Shut down the system.

```
sudo systemctl poweroff
```

19. Disable `Adapter 2` in the network settings.

[10] Using real hardware the network interface names may differ from the ones preconfigured in `/etc/netplan/00-installer-config.yaml` and must be manually adjusted.

Router

1. Clone the base VM image.
2. Name the cloned VM `Isolated and Exhausted Router`.
3. Change the VM properties to use 2 CPU cores, 1 GiB RAM.
4. Add three `Internal Network` adapters to the VM and assign the following network names:
    - Adapter1: `victim_provider`
    - Adapter2: `attacker_provider`
    - Adapter3: `dns_provider`
5. Boot the system and enable IPv6 forwarding

```
sudo sysctl -w net.ipv6.conf.all.forwarding=1
```

6. Change the hostname to `si-router`.

```
sudo bash -c "echo si-router > /etc/hostname"
```

7. Copy the configuration to the router VM[10]

```
sudo rm -f /etc/netplan/*
sudo cp -r $ARTIFACTS_REPO/hosts/router/rootfs/* /
```

8. Reboot.

```
sudo systemctl reboot
```

9. Enable and start the traffic shaping service.

```
sudo systemctl enable traffic-shaping.service
sudo systemctl start traffic-shaping.service
```

10. Check the MAC addresses of the interfaces `enp0s3` (`victim_provider`), `enp0s8` (`attacker_provider`) and `enp0s9` (`dns_provider`) and ensure that they are assigned to the corresponding VirtualBox adapters.
11. Shut down the system.

```
sudo poweroff
```

DNS Server

1. Clone the base VM image.
2. Name the cloned VM `Isolated and Exhausted DNS`.
3. Change the VM properties to use 2 cores, and 1 GiB RAM.
4. Boot the system and change the hostname to `si-dns`:

```
sudo bash -c "echo si-dns > /etc/hostname"
```

5. Install `bind9`.

```
sudo apt install bind9
```

6. Copy the configuration to the router VM[10]

```
sudo rm -f /etc/netplan/*
sudo cp -r $ARTIFACTS_REPO/hosts/dns/rootfs/* /
```

7. Disable `systemd-resolved`.

```
sudo systemctl disable systemd-resolved
```

8. Power off the system.

```
sudo systemctl poweroff
```

9. Change the VM network settings
   - Right-click the DNS VM in the VirtualBox Manager.
   - Select the option `Settings` from the drop-down menu.
   - Make sure that `Adapter 1` is enabled and that all other adapters are disabled.
   - For `Adapter 1` set the network type to `Internal Network` named `dns_provider`.

Victim VM Create a Virtual Box VM with 4 CPU cores and 8 GiB RAM and a 50 GiB hard disk or use an equivalent physical host. Note: Windows, Chrome and Firefox may perform fully automatic updates without prompting the user when an internet connection is available. Performing the victim installation offline solves this problem. The dependencies listed in subsubsection A.2.4 can be transferred to the victim VM via a shared folder, or via USB stick in case dedicated physical machines are used for the experiment.

1. Change the VM network settings.
   - Right-click the Victim VM in the VirtualBox Manager
   - Select the option `Settings` from the drop-down menu.
   - Make sure that Adapter 1 is enabled and that all other adapters are disabled
   - Attach Adapter 1 to an Internal Network named `victim_provider`.
   - Set the adapter type to `Intel PRO/1000 MT Desktop (82540EM)`.
2. Install Windows 10.
3. Install Chrome 83.0.4103.106 and Firefox Nightly 86.01a. Note: `chrome.exe` is expected to be located at `C:\ProgramFiles(x86)\Google\Chrome\Application\chrome.exe`. If chrome is installed elsewhere, the content of the variable `ATTACK_CMD` in `attack_simulator.py`, line 14 must be adjusted to point to `chrome.exe`.
4. Install the CA certificate from the artifact repository in Chrome:
   - Open Chrome and navigate to `chrome://settings/privacy`.
   - In the `Privacy and security` box click on `More`.
   - Click `Manage Certificates`.
   - Once the `Certificate Import Wizard` opened click `Next`.
   - Select the CA certificate from `isolated-and-exhausted/hosts/attacker/webserver/rootfs/srv/ca/ca.crt.pem` and proceed by clicking `Next`.
   - In the `Certificate Store` dialog click `Browse...` and change the Certificate Store to `Trusted Root Certification Authorities`
   - Proceed by clicking `Next` and the finish the import by clicking `Finish`.
5. Install the CA certificate from the artifact repository in FireFox Nightly:
   - Open Firefox and navigate to `about:preferences#privacy`.
   - Scroll down to the option group labeled `Security`.
   - Click on `View Certificates`
   - In the `Certificate Manager` Dialog select the `Autorities` tab and click `Import`.
   - Select the CA certificate from `isolated-and-exhausted/hosts/attacker/webserver/rootfs/srv/ca/ca.crt.pem` and proceed by clicking `Open`.
   - In the `Downloading Certificate` Dialog check `Trust this CA to identify websites.` and click `OK`.
6. Install Python 3.8.2
7. Install the Python Websocket Client (cf. subsubsection A.2.4)
8. Copy `websocket_client-1.3.2-py3-none-any.whl` to `C:\Users\user\Downloads`.
9. Open `cmd.exe` and execute the following commands to install the Python Websocket Client:

```
c:
cd \Users\user\Downloads
pip install websocket_client-1.3.2-py3-none-any.
    ↪ whl
```

10. Open the Windows `Network Connections` dialog via `Startmenu → Settings → Network & Internet → Ethernet → Change adapter options`.
11. Right-click the network connection (e.g. `Ethernet`).
12. Select `Properties` from the drop-down menu.
13. In the `Ethernet Properties` dialog select `Internet Protocol Version 6 (TCP/IPv6)`
14. Click `Properties`.
15. Adjust the Windows network adapter settings:
    - IPv6 Address: `2001:db8::4000:1`
    - Subnet prefix length: `98`
    - Default gateway: `2001:db8::7fff:ffff`
    - DNS server: `2001:db8::8000:1`
16. Confirm the changes by clicking `OK`.
17. Copy the artifact repository folder `isolated-and-exhausted` to `C:\Users\user\Documents`.

Chrome Build Environment This paragraph describes how to set up a build environment for Chromium and is mostly based on the official Chromium Build Instructions.[11]

1. Install Kubuntu 20.04 LTS on a host with at least 16 GiB of RAM and at least 300 GiB of free hard disk space.
2. Insert the VirtualBox guest additions ".iso" into the virtual CD-ROM drive of your machine.
3. Install the VirtualBox guest additions.

```
sudo mount /dev/sr0 /media
cd /media
sudo ./VBoxLinuxAdditions.run
```

4. Add `user` to the `vboxsf` group

```
sudo usermod -a -G vboxsf user
```

5. Install additional required packages.

```
sudo apt install git build-essential
```

6. Reboot the system.

```
sudo reboot
```

7. Clone the Chromium depot tools and add their path to the `PATH` environment variable:

```
git clone https://chromium.googlesource.com/
    ↪ chromium/tools/depot_tools.git
export PATH="$PATH:/home/user/depot_tools"
echo export PATH=\"\$PATH:/home/user/depot_tools\"
    ↪  >> ~/.bash_profile
```

8. Get the Chromium source code (this may take a while).

```
mkdir ~/chromium && cd ~/chromium
fetch --nohooks chromium
cd src
./build/install-build-deps.sh
git fetch --tags
git checkout tags/101.0.4951.64
```

[11] https://chromium.googlesource.com/chromium/src/+/main/docs/linux/build_instructions.md

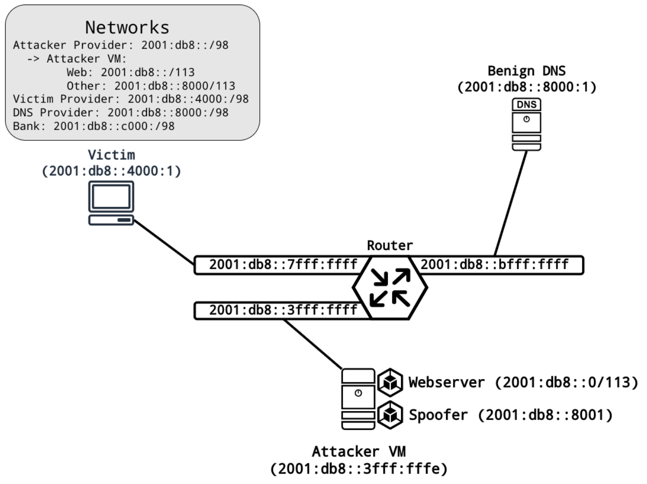
Networks
Attacker Provider: 2001:db8::/98
-> Attacker VM:
Web: 2001:db8::/113
Other: 2001:db8::8000/113
Victim Provider: 2001:db8::4000:/98
DNS Provider: 2001:db8::8000:/98
Bank: 2001:db8::c000:/98
Benign DNS (2001:db8::8000:1)
DNS
Victim (2001:db8::4000:1)
Router
2001:db8::7fff:ffff
2001:db8::bfff:ffff
2001:db8::3fff:ffff
Webserver (2001:db8::0/113)
Spoofer (2001:db8::8001)
Attacker VM (2001:db8::3fff:fffe)

Figure 1: Network configuration of the Isolated and Exhausted evaluation lab.

9. Prepare the build. The command `gn args ...` automatically opens a file in the default text editor. Replace the contents of this file with the contents of the file `~/isolated-and-exhausted/site_isolation_patch/gn.args` from the artifacts repository.

```
gclient sync -D --with_branch_heads
gclient runhooks
gn args out/release
```

10. Finalize build preparations.

```
gn gen out/release
```

11. Build an unmodified Chrome (this may take a while).

```
nice -n 19 autoninja -C out/release
```

12. Build a version of Chrome patched with our proof-of-concept mitigation against the Site Isolation fork-bomb (this may take a while). Pass the contents of `~/isolated-and-exhausted/site_isolation_patch/gn.args` as arguments when invoking `gn args`.

```
git apply ~/isolated-and-exhausted/
    ↪ site_isolation_patch/si_patch.diff
gn args out/patched
gn gen out/patched
nice -n 19 autoninja -C out/patched
```

A.3.2 Basic Test

Testing Connectivity After successfully performing the setup (cf. subsection A.3) the lab network should be configured as shown in Figure 1. To verify the setup make sure that all hosts can communicate with each other by issuing `ping` commands. Note that Windows 10 will not respond to incoming ICMP echo requests but should be able to receive ICMP echo responses from all other hosts in reaction to requests sent from the victim system. If mutual communication between all hosts works, ensure that DNS resolution works on the victim system. Open the windows command prompt (`cmd.exe`) and issue the command `nslookup evil.com`. You should receive a response resolving the domain to the IP address `2001:db8::1`.

Testing the Attacker Website To test the attacker web server, open a browser on the victim host and navigate to `http://evil.com`. You should see the website shown in Figure 2. If you properly installed the root CA certificate from the Isolated and Exhausted repository you should also be able to access the same website via HTTPS on port 443 without receiving a self-signed certificate warning message.

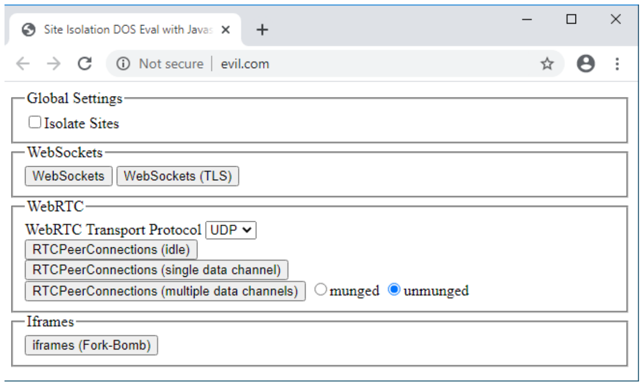
Site Isolation DOS Eval with Java
Not secure | evil.com
Global Settings
Isolate Sites
WebSockets
WebSockets
WebSockets (TLS)
WebRTC
WebRTC Transport Protocol UDP
RTCPeerConnections (idle)
RTCPeerConnections (single data channel)
RTCPeerConnections (multiple data channels)
munged
unmunged
Iframes
iframes (Fork-Bomb)

Figure 2: A website to test Site Isolation exploits. The website can be used to allocate UDP ports using WebRTC and to execute a Site Isolation fork-bomb.

Testing the Automated DEMONS Experiment Open an instance of the *Task Manager*, select the performance tab and focus the CPU graph. Next, open the windows command line (`cmd.exe`) and change your working directory to the victim host folder inside the Isolated and Exhausted artifact repository and execute the `attack_simulator.py` script.

```
c:
cd \Users\user\Documents\isolated-and-exhausted\hosts\
    ↪ victim
python attack_simulator.py
```

Once the `attack_simulator.py` script is running, an instance of the configured browser (or the malware attacker) should be started. In the Task Manager, you should be able to observe the CPU load spiking for a couple of seconds at the same time, the number of open handles will rise to a value around 130000 but little to no network traffic is observable during the DEMONS setup phase. Once the setup phase completes, CPU load will drop significantly. At the same time spoofed DNS responses sent by the poisoner consume a moderate amount of downstream bandwidth.

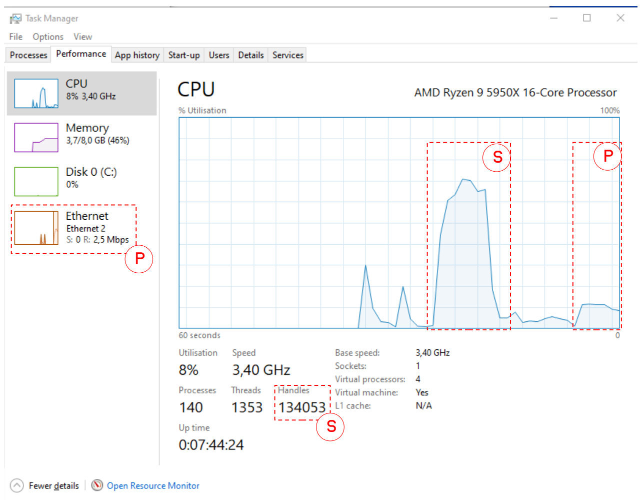

Figure 3: During the DEMONS Setup Phase (S) the CPU load and number of handles increase significantly, the Poisoning Phase (P) causes moderate CPU and network load.

Testing the Custom Chromium Build. Boot the Chromium Build Environment, open a console and change your working directory to the subfolder `src` inside the chromium repository. Then run both the unpatched and the patched version of Chromium. In both cases, you should be presented with a Chromium browser window. Make sure that you can access the Chromium developer tools.

```
cd ~/chromium/src
out/release/chrome
out/patched/chrome
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The Site Isolation fork-bomb can be used to implement DoS attacks against the operating system or the web browser.

(C2): Currently none of the tested browsers mitigates the Site Isolation fork-bomb. We propose an effective mitigation and implemented a proof-of-concept patch based on Chromium 101.0.4951.64. Our patch is efficient and does not measurably affect the browser's performance.

(C3): The impact of Site Isolation attacks goes beyond DoS. ***D****NS-Poisoning by* ***E****xhaustive* ***M****isappropriation* *of* ***N****etwork* ***S****ockets (DEMONS)*, a DNS Cache Poisoning attack uses Site Isolation to poison the DNS cache of the Windows operating system, in the web attacker model.

A.4.2 Experiments

(E1): [Site Isolation Fork-Bomb (ad C1)] [1 human-hour]

```
START;END;RESULT;BURSTS;DURATION;CACHE_CONTENT
2022-09-19 11:52:20.508692;2022-09-19 12:02:23.901948;SUCCESS;5779;603;2001:db8::1|2001:db8::1
2022-09-19 12:02:33.976641;2022-09-19 12:36:50.096716;SUCCESS;20146;2056;2001:db8::1|2001:db8::1
2022-09-19 12:37:00.154377;2022-09-19 12:40:34.654990;SUCCESS;1998;214;2001:db8::1|2001:db8::1
2022-09-19 12:40:44.747582;2022-09-19 13:25:14.253180;ABORT;25000;2669;None
...
```

Listing 1: Example excerpt of the DEMONS result log on the attacker server. Each line represents an iteration of the DEMONS experiment against the victim host and contains the following values in order from left to right: start date of the experiment end date of the experiment, outcome, duration of the experiment in seconds, the IP address found in the victim's cache for the target domain.

Preparation:

1. Boot the router, the benign DNS, the attacker server and the victim host.
2. Log into the victim system.
3. Start the Windows TaskManager and monitor the browser processes on the "Processes" tab.

Execution: Open Chrome 83.0.4103.106 and perform the following steps:

1. Visit the attacker web site in the lab http://evil.com:80 (see Figure 2).
2. Note the number of browser processes running
3. Make sure the checkbox `Isolate Sites` is unchecked.
4. Click the button labeled `iframes (Fork-Bomb)`
5. Note that the number of browser processes has not changed significantly.
6. Make sure the checkbox `Isolate Sites` is checked.
7. Click the button labeled `iframes (Fork-Bomb)`
8. Observe the number of browser processes increase significantly until it stalls after some time.
9. If necessary kill the browser process or reset the victim system.

Results: Once the victim system is no longer able to create new browser processes the browser may crash and/or the victim OS becomes unusable. For a detailed description of effects we observed during our evaluation please refer to Table 5 in Appendix C of our work.
The number of processes created depends on the hardware, host OS, browser, browser version, and swap configuration. Even using identical hard- and software, the number of processes varies between runs. Table 2 of in our work lists the median of five measurements.

(E2): *[Site Isolation Fork-Bomb mitigation (ad C2)] [1 human-hour]*

Preparation: Start the Chrome Build Environment (cf. section A.3.1 Chrome Build Environment)

Execution: To verify that the changes introduced by our patch do not measurably impact the browser's performance, record the page load times of the Tranco-Top 5 websites using the profiling tools integrated into Chromium[12] (see Table 1).
Create at least one series of measurements for each Chromium and Chromium-Site Isolation-patched[13] by performing the following steps:

- Start the browser.
- Open the performance tab of the integrated developer tools.
- Open the website.
- Run the profiler and discard the initial result to avoid any caching effects.
- Run the profiler and calculate $t_{load} = t_{total} - t_{idle}$, where t_{total} is the total time recorded by the profile and t_{idle} is the time the browser was idle during loading.
- Repeat the previous step four more times.

Website	Page load time in ms Chromium 101.0.4951.64	
	unpatched	patched
google.com	752.1 ms	766.4 ms
youtube.com	5366.4 ms	5265.5 ms
facebook.com	643.3 ms	629.5 ms
netflix.com	1107.3 ms	1039.6 ms
microsoft.com	1305 ms	1243.3 ms

Table 1: The average page load time (in ms) of the Tranco-Top-5 websites shows no significant difference between an unpatched Chromium 101.0.4951.64 and the same browser version patched with our mitigation against the Site Isolation fork-bomb.

Results: Table 1 shows the average loading times for the Tranco-Top-5 websites using Chromium and Chromium Site Isolation-patched. There is no significant difference in performance between both browser versions (see `isolated-and-exhausted/site_isolation_patch/si_patch_performance_t_test.ods`). Since we only need two additional global constants, and one additional

[12] https://developer.chrome.com/docs/devtools/evaluate-performance/?utm_source=devtools

[13] To determine the page load time we recorded two series of measurements on two different days, to reduce the impact of server and network load.

local limit per tab/window, the effect of our patch on browser memory consumption is negligible.

(E3): [DNS Poisoning by Exhaustive Misappropriation of Network Sockets (DEMONS) (ad C3)] [5 human minutes + 12 compute hours]
A victim visits the attacker web site and becomes subject to the DEMONS (attack), if successful poisons the victim's DNS cache. A script automatically repeats the experiment until it is stopped manually.

Preparation:

1. Boot the router, benign DNS, attacker server and the victim host.
2. Log into a shell on the attacker server.
3. Start live monitoring of DEMONS experiment results (cf. Listing 1).

```
cd /srv/docker/data/demons/attacker/spoofer/
    ↪ results/
tail -f $(ls -1r | head -n1)
```

Execution:

1. Open `cmd.exe` on the victim system.
2. Change your working directory to the victim host sub folder in the artifacts repository and run the experiment:

```
c:
cd \Users\user\Documents\isolated-and-
    ↪ exhausted\hosts\victim
python attack_simulator.py
```

Results: DEMONS is probabilistic because the attacker must correctly guess a random 16-bit transaction ID. Given a large enough sample set, the attacker can poison the victim's DNS cache with a success probability of 36% or better. Each DEMONS experiment may have one of three results `SUCCESS` in case the attacker successfully poisoned the victim's DNS cache, `FAILURE` - a DNS response from the benign DNS server was cached by the victim or `ABORT` if neither the attacker nor the benign DNS served a valid record within a preset burst limit.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220912. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenix%20sec2023/.